

GWU2X プログラミング ガイド_U2X_IIC

UG1002-1.0J, 2021-06-29

著作権について（2021）

著作権に関する全ての権利は、**Guangdong Gowin Semiconductor Corporation** に留保されています。

GOWIN高云、Gowin、及びGOWINSEMIは、当社により、中国、米国特許商標庁、及びその他の国において登録されています。商標又はサービスマークとして特定されたその他全ての文字やロゴは、それぞれの権利者に帰属しています。何れの団体及び個人も、当社の書面による許可を得ず、本文書の内容の一部もしくは全部を、いかなる視聴覚的、電子的、機械的、複写、録音等の手段によりもしくは形式により、伝搬又は複製をしてはなりません。

免責事項

当社は、**GOWINSEMI Terms and Conditions of Sale**（GOWINSEMI取引条件）に規定されている内容を除き、（明示的か又は黙示的に拘わらず）いかなる保証もせず、また、知的財産権や材料の使用によりあなたのハードウェア、ソフトウェア、データ、又は財産が被った損害についても責任を負いません。本文書における全ての情報は、予備的情報として取り扱われなければなりません。当社は、事前の通知なく、いつでも本文書の内容を変更することができます。本文書を参照する何れの団体及び個人も、最新の文書やエラッタ（不具合情報）については、当社に問い合わせる必要があります。

バージョン履歴

日付	バージョン	説明
2021/06/29	1.0J	初版。

目次

目次.....	i
図一覧	ii
表一覧	iii
1 機能の紹介	1
2 ドライバーのインストールとアンインストール.....	2
2.1 Zadig を使用してドライバーをインストール	2
2.2 ドライバーのアンインストール	4
3 libusb_WinUSB プログラミングの説明.....	6
3.1 libusb 関数ライブラリの初期化と終了	6
3.2 指定された USB デバイスを開く	7
4 U2X_IIC の API 関数.....	9
4.1 U2X_IIC の初期化	9
4.2 データの送信	9
4.3 インターフェース宣言.....	10
4.4 データの受信	11
4.5 プログラミング例.....	11
5 エラーコード.....	14
用語、略語.....	16
テクニカル・サポートとフィードバック	17

図一覧

図 2-1“List All Device”オプションを選択.....	2
図 2-2 デバイスを選択	3
図 2-3 ドライバーを選択.....	3
図 2-4 デバイスマネージャーを開く	4
図 2-5 ドライバーのアンインストール.....	5

表一覧

表 5-1 エラーコード一覧.....	14
表 A-1 用語、略語	16

1 機能の紹介

GWU2X は、USB2IIC 変換を実現できる USB からマルチプロトコルへのコンバータであり、最大 500KHz（現在のテスト結果）の SCL クロック周波数をサポートします。

IIC マスターモードのデータ送受信機能をサポートし、7 ビットアドレスモードと 10 ビットアドレスモードをサポートします。

2 ドライバーのインストールとアンインストール

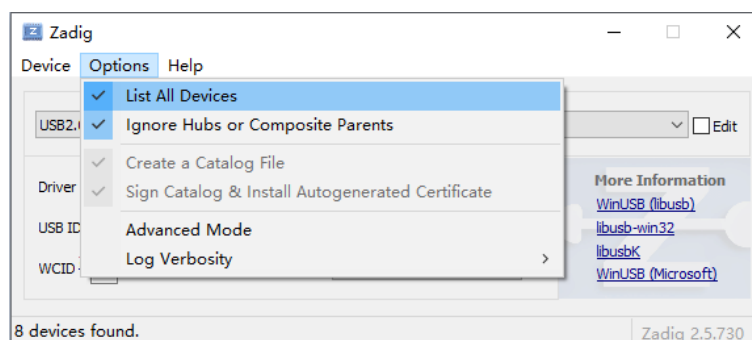
オープンソースの USB 関数ライブラリ libusb を使用して、GWU2X をプログラミングすることができます。この関数ライブラリを使用してプログラミングする場合は、WINDOWS での USB ドライバーの WinUSB.sys をインストールする必要があります。

ドライバーをインストールするにはオープンソースのドライバー・インストー・ルツール Zadig (<https://zadig.akeo.ie/>) を使用できます。ドライバーをインストールするには、管理者権限が必要です。

2.1 Zadig を使用してドライバーをインストール

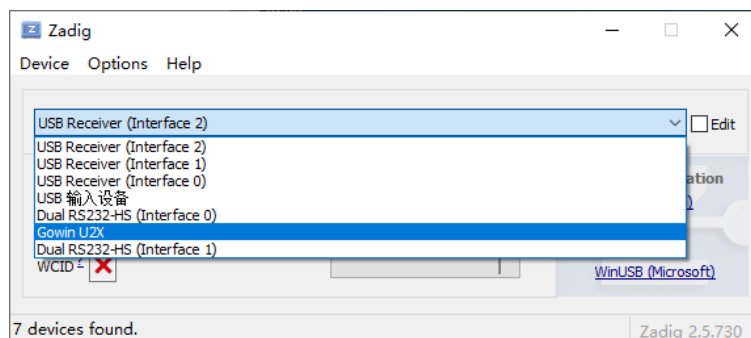
GWU2X デバイスをコンピューターの USB インターフェースに接続し、Zadig をダブルクリックして開き（管理者権限が必要）、Options > List All Device をチェックすると、コンピュータに接続されているすべての USB デバイスが一覧表示されます。

図 2-1 “List All Device” オプションを選択



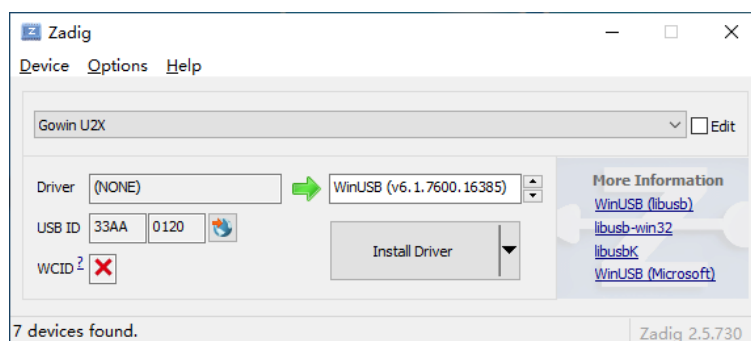
デバイス GWU2X を選択して、ドライバーをインストールします。

図 2-2 デバイスを選択



インストールするドライバーを選択します。libusb + WinUSB を使用する
場合、WinUSB を選択してください。

図 2-3 ドライバーを選択



“Install Driver”^[1] ボタンをクリックして、ドライバーをインストールし
ます。

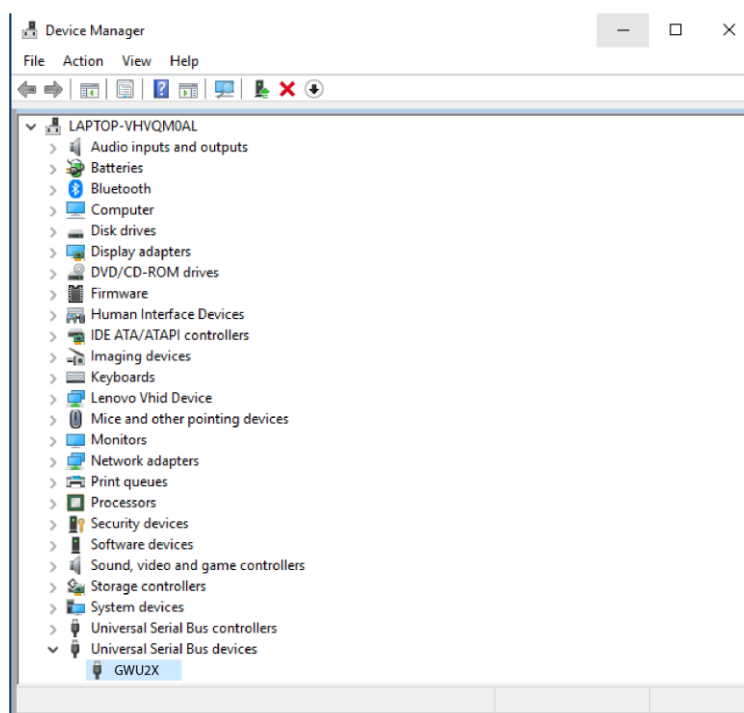
注記：

ドライバーが現在インストールされていない場合は“Install Driver”、他のドライバーがイ
ンストールされている場合は“Replace Driver”と表示されます。

2.2 ドライバーのアンインストール

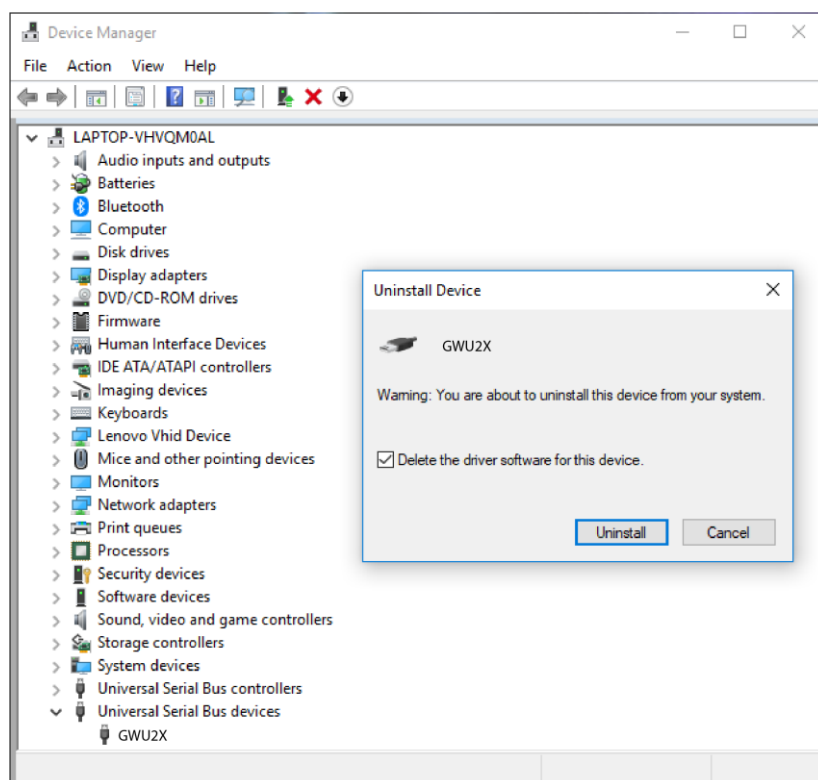
ドライバーをアンインストールするときは、GWU2X デバイスをコンピューターに接続し、Windows デバイスマネージャーを開いて、“ユニバーサル シリアル バス デバイス” リストで GWU2X デバイスを特定します。デバイス名を右クリックして、右クリックメニューの“削除”オプションを選択します。

図 2-4 デバイスマネージャーを開く



ポップアップダイアログボックスで、“このデバイスのドライバー ソフトウェアを削除する”にチェックを入れてアンインストールします。

図 2-5 ドライバーのアンインストール



3 libusb_WinUSB プログラミングの説明

libusb はオープンソースの USB 関数ライブラリです。

その公式ウェブサイトは <https://libusb.info> です。

そのソースコードは <https://github.com/libusb/libusb> から入手できます。

動的ライブラリと静的ライブラリを含む、コンパイル済みバージョン、公式 GCC バージョン、および VS バージョンは、公式 Web サイトからダウンロードできます。github からソースコードをダウンロードして、自分でコンパイルすることもできます。

libusb 関数宣言については、公式リファレンス <http://libusb.sourceforge.net/api-1.0> を参照してください。

3.1 libusb 関数ライブラリの初期化と終了

libusb を使用してプログラミングする場合は、関数 `libusb_init ()` を呼び出して初期化する必要があります。使用後は、関数 `libusb_exit ()` を呼び出して終了させる必要があります。

関数宣言は次のとおりです。

```
int libusb_init (libusb_context ** context)

void libusb_exit (libusb_context * ctx)
```

パラメータ `libusb_context` は libusb コンテキスト構造体であり、libusb のいくつかの構成パラメータを保存するために使用されます。libusb_context が指定されていない場合、デフォルトのコンテキスト構造が作成されます。コンテキスト構造がすでに存在する場合、その構造は再初期化せずに直接使用されます。

プログラミング例：

```
int rc = libusb_init(NULL);

if (rc < 0)
```

```
    return rc;

    libusb_exit(NULL);
```

3.2 指定された USB デバイスを開く

`libusb_open_device_with_vid_pid()` 関数を使用して、VID/PID に従って指定されたデバイスを開くことができます。`libusb_get_device_list()` 関数を使用してすべての USB デバイスを取得し、それらから必要なデバイスを選択し、`libusb_open()` 関数を使用して後続の操作のためにデバイスのハンドルを取得することもできます。関数の説明は次のとおりです。

VID/PID を使用してデバイスを開きます。

```
libusb_device_handle* libusb_open_device_with_vid_pid(

    libusb_context * ctx,

    uint16_t vendor_id,

    uint16_t product_id

);
```

その中で、パラメータ `ctx` は、`libusb` が初期化されるときに生成されるコンテキスト構造のアドレスです。デフォルトのコンテキストが使用される場合、`NULL` が使用されます。`vendor_id` と `product_id` は、それぞれ USB デバイスの VID と PID です。Gowin USB デバイスの VID は `0x33aa` であり、USB2IIC デバイスの PID は `0x0020` です。

返り値は、このコンピュータで `libusb` によって最初に一致したデバイスの操作ハンドルポインターです。それ以外の場合は、`NULL` ポインタが返されます。

使用例は次のとおりです。

```
devh = libusb_open_device_with_vid_pid(NULL, 0x33aa, 0x0020);

if(NULL == devh) {

    printf("Open USB device failed\n");

    goto out;

}
```

すべての USB デバイスを取得した後、指定したデバイスを選択します。

```
ssize_t libusb_get_device_list (

    libusb_context *ctx,

    libusb_device *** list

)
```

その中で、パラメータ **ctx** は、**libusb** が初期化されるときに生成されるコンテキスト構造のアドレスです。デフォルトのコンテキストが使用される場合、**NULL** が使用されます。**list** はデバイスリストへのポインタです。

使用後、**libusb_free_device_list ()** 関数を使用してそのメモリを解放する必要があります。

関数が正しく実行された場合、返り値はデバイスの数であり、見つかったデバイスのリストが **list** に保存されます。それ以外の場合は、ゼロ未満の **libusb_error** 値が返されます。

```
int libusb_open (
    libusb_device *dev,
    libusb_device_handle **dev_handle
)
```

その中で、パラメータ **dev** はデバイスリスト内のデバイスであり、**dev_handle** は返されるデバイスハンドルへのポインタのアドレスです。

実行が成功した場合、**0** が返されます。成功しなかった場合、ゼロ未満の **libusb_error** 値が返されます。

使用例は次のとおりです。

```
cnt = libusb_get_device_list(NULL, &devs);

if(cnt < 0) {
    // get device list failed
    return -1;
}

for(int i = 0; i < cnt; i++) {
    libusb_open(dev[i], dev_handle);

    if(/*the wanted device is opened*/) {
        break;
    } else {
        //the current device is not wanted, close it and check the next one.
        libusb_close(dev_handle);
    }
}
```

4 U2X_IIC の API 関数

4.1 U2X_IIC の初期化

この関数は、U2X_IIC の初期化や、SCL クロック周波数の構成に使用されます。SCL クロック周波数を変更する必要がある場合は、構成のためにこの関数を再度呼び出す必要があります。

```
int u2x_iic_init(  
  
libusb_device_handle *devh,  
  
unsigned int uiFreqKiloHz,  
  
unsigned int uiTimeout);
```

パラメータ :

- devh : libusb のデバイス操作ハンドル。
- uiFreqKiloHz : SCL クロックの周波数 (KHz) 。
- uiTimeout : タイムアウトパラメータ。単位はミリ秒です。

返り値 :

実行が成功した場合、0 が返されます。成功しなかった場合、ゼロ未満のエラーコードが返されます。

4.2 データの送信

IIC を介して指定された IIC スレーブデバイスにデータを送信します。1 回の送信の最大データ長は 256 バイトです。

```
int u2x_iic_send_bytes(  
  
libusb_device_handle *devh,  
  
unsigned int uiAddr,  
  
unsigned int uiAddrBit,
```

```
unsigned char *pucData,  
  
unsigned int uiDataLen,  
  
unsigned int uiTimeout);
```

パラメータ :

- **devh** : libusb のデバイス操作ハンドル。
- **uiAddr** : IIC スレーブデバイスのアドレス。
- **uiAddrBit** : アドレスビット数、7 ビットアドレスを使用する場合、このパラメータを 7 に設定します。10 ビットアドレスを使用する場合、このパラメータを 10 に設定します。他の値は使用できない値です。
- **pucData** : 送信されるデータが保存されているメモリアドレスを指すポインタ。
- **uiDataLen** : 送信されるデータのバイト数
- **uiTimeout** : タイムアウトパラメータ。単位はミリ秒です。

返り値 :

実行が成功した場合、0 が返されます。成功しなかった場合、ゼロ未満のエラーコードが返されます。

4.3 インターフェース宣言

USB デバイスには通常、1 つ以上のインターフェースが含まれています。libusb がインターフェースを使用する場合、最初にインターフェースを宣言 (claim interface) する必要があります。宣言が成功すると、インターフェースが正常に開かれ、インターフェースに含まれるエンドポイントに対して受送信できることになります。

```
int libusb_claim_interface(  
  
    libusb_device_handle * dev_handle,  
  
    int interface_number  
  
)
```

その中で、パラメータ **dev_handle** はデバイスハンドルであり、**interface_number** はインターフェース番号です。GWU2X デバイスでは、インターフェース番号は 0 です。インターフェースの宣言が成功した場合は 0 が返され、そうでない場合はゼロ未満の **libusb_error** 値が返されます。

プログラミング例

```
rc = libusb_claim_interface(devh, 0);  
  
if (rc < 0) {
```

```
printf("Error claiming interface: %s\n", libusb_error_name(rc));

goto out;

}
```

4.4 データの受信

IIC を介して指定された IIC スレーブデバイスからデータを受信します。1 回の受信の最大データ長は 256 バイトです。

```
int u2x_iic_read_bytes(

libusb_device_handle *devh,

unsigned int uiAddr,

unsigned int uiAddrBit,

unsigned char *pucData,

unsigned int uiDataLen,

unsigned int uiTimeout);
```

パラメータ :

- **devh** : libusb のデバイス操作ハンドル。
- **uiAddr** : IIC スレーブデバイスのアドレス。
- **uiAddrBit** : アドレスビット数、7 ビットアドレスを使用する場合、このパラメータを 7 に設定します。10 ビットアドレスを使用する場合、このパラメータを 10 に設定します。他の値は使用できない値です。
- **pucData** : 受信されるデータが保存されているメモリアドレスを指すポインタ。
- **uiDataLen** : 受信されるデータのバイト数
- **uiTimeout** : タイムアウトパラメータ。単位はミリ秒です。

返り値 :

実行が成功した場合、0 が返されます。成功しなかった場合、ゼロ未満のエラーコードが返されます。

4.5 プログラミング例

```
int main(int argc, char *argv[])

{

unsigned char txdata[TEST_BYTE];

unsigned char rxdata[TEST_BYTE];
```

```
int i = 0;

int rc = 0;

// デバイスを開きます

rc = libusb_init(NULL);

if (rc < 0)

return rc;

devh = libusb_open_device_with_vid_pid(NULL, 0x33aa, 0x0120);

if(NULL == devh) {

printf("Open USB device failed¥n");

goto out;

}

rc = libusb_claim_interface(devh, 0);

if (rc < 0) {

printf("Error claiming interface: %s¥n", libusb_error_name(rc));

goto out;

}

txdata[0] = 0x55;

for(i = 1; i < TEST_BYTE; i++) {

txdata[i] = i;

}

// U2X IIC 初期化

u2x_iic_init(devh, 400, 1000);

// アドレス 0x50 の IIC スレーブデバイスに 4 バイトのデータを送信します。アドレスは 7 ビットモードです。

u2x_iic_send_bytes(devh, 0x50, 7, txdata, 4, 1000);

// アドレス 0x50 の IIC スレーブデバイスから 4 バイトのデータを読み出します。アドレスは 7 ビットモードです。

u2x_iic_read_bytes(devh, 0x50, 7, rxdata, 4, 1000);

// デバイスを閉じて終了します

libusb_release_interface(devh, 0);

out:
```

```
    if(devh)

    libusb_close(devh);

    libusb_exit(NULL);

    return 0;

}
```

5 エラーコード

API 関数が正常に実行されている場合、戻り値は **0** です。それ以外の場合は、負の数を返します。

0 以外の戻り値で表されるエラーの意味を次の表に示します。

表 5-1 エラーコード一覧

Value	Enumerator	
0	SUCCESS	エラーなし
-1	USB_ERROR_IO	USB 入出力エラー
-2	USB_ERROR_INVALID_PARAM	USB パラメータエラー
-3	USB_ERROR_ACCESS	デバイスにアクセスするための権限が不十分です
-4	USB_ERROR_NO_DEVICE	USB デバイスが見つかりません(デバイスが切断されています)
-5	USB_ERROR_NOT_FOUND	エンティティ (Entity) が見つかりません
-6	USB_ERROR_BUSY	USB デバイスがビジーです
-7	USB_ERROR_TIMEOUT	タイムアウト
-8	USB_ERROR_OVERFLOW	メモリーオーバーフロー
-9	USB_ERROR_PIPE	パイプエラー
-10	USB_ERROR_INTERRUPTED	システム関数が中断されました
-11	USB_ERROR_NO_MEM	メモリ不足
-12	USB_ERROR_NOT_SUPPORTED	この操作は現在のプラットフォームではサポートされていません
-13	U2X_IIC_ERROR_USBTRANS_ERR	USB データ転送エラー
-14	U2X_IIC_ERROR_INVALID_PARAM	パラメータ無効

Value	Enumerator	
-15	U2X_IIC_ERROR_TIMEOUT	U2X_IIC 転送タイムアウト
-16	U2X_IIC_ERROR_CMD_ERR	U2X_IIC 命令エラー
-17	U2X_IIC_ERROR_NO_SPEC_SLV	U2X_IIC は指定されたスレーブデバイスを見つけられませんでした
-99	ERROR_OTHER	その他のエラー

用語、略語

表 A-1 に、本マニュアルで使用される用語、略語、及びその意味を示します。

表 A-1 用語、略語

用語、略語	正式名称	意味
USB	Universal Serial Bus	ユニバーサル・シリアル・バス
IIC	Inter—Integrated Circuit	2 線式シリアルバス

テクニカル・サポートとフィードバック

GOWIN セミコンダクターは、包括的な技術サポートをご提供しています。使用に関するご質問、ご意見については、直接弊社までお問い合わせください。

Web サイト : www.gowinsemi.com

E-mail : support@gowinsemi.com

